A Genetic Network Programming Based Method to Mine Generalized Association Rules with Ontology

Guangfei Yang, Kaoru Shimada, Shingo Mabu, Kotaro Hirasawa, and Jinglu Hu

Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka 808-0135, Japan
E-mail: yang@fuji.waseda.jp, k.shimada@aoni.waseda.jp, {mabu, hirasawa, jinglu}@waseda.jp
[Received April 27, 2007; accepted August 31, 2007]

In this paper, we propose a Genetic Network Programming based method to mine equalized association rules in multi concept layers of ontology. We first introduce ontology to facilitate building the multi concept layers and propose Dynamic Threshold Approach (DTA) to equalize the different layers. We make use of an evolutionary computation method called Genetic Network Programming (GNP) to mine the rules and develop a new genetic operator to speed up searching the rule space.

Keywords: generalized association rule, genetic network programming, ontology, dynamic threshold approach

1. Introduction

Many studies have been done for mining the association rules efficiently. Most of the association rules mining methods describe the items in the rules by some simple keywords. For example, they could find that 66.7% customers who buy women’s dress may also purchase women’s dress shoes. In this rule, women’s dress and women’s dress shoes are keywords. One concept level rules mining [1] or leaf-level rules mining [2] features these methods.

Association rules in multi concept layers, also known as generalized association rules, or multiple-level association rules, could provide more information at multiple concept levels [1, 2]. We will use the phrase generalized association rules in the following of this paper. For the above example, generalized association rules could tell us the purchasing relationship not only between women’s dress and women’s dress shoes, but also the relationship including women’s clothing and women’s shoes, which are more generalized concepts. In Han and Fu’s method [1], a taxonomy should be first defined, and rules are discovered only between concepts within the same layer of the hierarchy. Srikant and Agrawal’s method [2] does not have this limitation, which could find rules among all the concepts in all layers. But these methods have the following shortcomings:

- Hard to build a taxonomy: A data mining expert usually needs the help of domain experts to build the taxonomy describing a certain domain. As a result, there is a new problem that they should be able to share the domain knowledge. The simple taxonomy is difficult to be reused since there are seldom standards for restricting the building process. Moreover, the simple structure could not describe the domain knowledge well.
- Hard to deal with inequality among layers: If we mine rules among all the concept layers, most items in rules are picked up only from the upper concept layers, because the frequencies of the concepts in these layers are much larger. If we consider these layers too much, we will lose much information between different layers.
- Wait for too many rules: The traditional methods usually could only give a few simple rules, e.g., with only one antecedent and one consequent item, during the early stage of the calculation process and result in too many rules after the whole calculations especially when dealing with large database.

In our research, we propose some new methods to solve these problems. There are three key features in our methods:

- Use ontology instead of taxonomy: We build and maintain an ontology instead of simple taxonomy by means of some free tools, therefore the semantic knowledge of the domain could be shared with different data mining experts.
- Equalize layers: We propose a Dynamic Threshold Approach to select a certain percentage of classes in each layer for rule mining by using different minimum support thresholds in different layers.
- Use evolutionary method: Genetic Network Programming is applied to mine rules generation by generation. Even when the database grows to a large size, we could get some rules after early generations.

The rest of this paper is organized as follows. In Section 2, we review the concepts of generalized association rule as well as ontology and extend generalized association rule by ontology. In section 3, we propose Dynamic
Threshold Approach to equalize the classes from different layers. In Section 4, we apply GNP in the form of some item chains to mine the ontology based generalized association rules and discuss new fitness functions and new genetic operators. In Section 5 we evaluate our method by four kinds of experiments. In Section 6, we give a summary of this paper.

2. General Architecture

In Fig. 1, we demonstrate the main components of our method [3]. Generally speaking, there are three parts: Database, Equalizer and GNP Miner. In Database, transaction records, ontology library and rule pool are stored. We first build the ontology to describe the classes and relations in a specific domain. In order to mine generalized association rules, the traditional simple transaction records should be transferred into Hierarchical Transaction. Protege accesses Database by JDBC API, while other modules connect Database by ODBC API. Because it is found that the classes in different layers have unequal chance to appear in the generalized association rules, we develop an Equalizer to equalize layers comparatively. Genetic Network Programming is introduced to mine the association rules and we employ a new genetic operator, named renewal, to explore the global rule space more efficiently. The details are explained in the later sections.

3. Ontology based Generalized Association Rules (OGAR)

3.1. Generalized Association Rules

Generalized association rules, also known as multiple-level association rules, could provide more information at multiple concept levels [1, 2]. The idea of hierarchy has been introduced into generalized association rules mining. In Han and Fu’s method [1], rules are discovered among concept items only within the same layer of the hierarchy. For example, in Fig. 2, they mine the rules among Men’s Clothing, Women’s Clothing, Teen Clothing, Men’s Shoes and Women’s Shoes. They do not discover the rules between different layers, e.g., rules between Women’s Clothing and Dress Shoes. As a result, some useful information among different concept layers would be missed. Srikant and Agrawal’s [2] method does not have this limitation, which could find rules among the concept items from all concept layers.

3.2. Ontology

There are various definitions of ontology. In computer science, it is used for describing the structure and processes of a specific domain. We summarize some definitions given by different experts [4–6].

- An ontology is a formal and explicit specification of shared conceptualization. (T. R. Gruber, 1995 [4])
- An ontology is a logic theory accounting for an intended meaning of a formal vocabulary. (N. Guarino, 1998 [5])
- Information system ontology is a formal language designed to represent a particular domain of knowledge. (G. L. Zuniga, 2001 [6])

Among the above definitions, the most frequently referenced one is given by T. R. Gruber and the formal notion of conceptualization is the most important and complex. A conceptualization is a formal structure of reality as perceived and organized by an agent which is independent of the vocabulary used and the actual occurrence of a specific situation. There are five main components in an ontology: class (also called concept), property (sometimes called slot or role) describing various features of classes, facet (sometimes called role restrictions) describing the restrictions on properties, instance and relationship [7].

We build a ontology with Protege [8] about the goods in a supermarket and Fig. 2 shows its fragment.

3.3. Steps to Build Ontology

There are mainly seven steps to build the ontology [9], see also Fig. 3:

- An ontology is a formal and explicit specification of shared conceptualization. (T. R. Gruber, 1995 [4])

Fig. 1. General architecture.
1. Determine the domain and scope of the ontology, and make a plan for the necessary tasks.

2. Analyze the requirements of the ontology, including goals, design guidelines, potential users, and so on. This step gives a semi-formal description of the ontology, which is a draft or outline to be detailed.

3. Consider reusing existing ontologies. Import some ontologies if necessary.

4. Formalize the semi-formal ontology in more details and provide a mature ontology.
   - Enumerate important terms in the ontology
   - Define the classes and class hierarchy
   - Define various relationships in the hierarchy
   - Define the properties of classes
   - Create instances

5. Evaluate the mature ontology according to the requirement specifications and the evaluation criteria to make sure of the usefulness of the developed ontology.

6. Write documents to record the details of the ontology clearly and exhaustively, including each phase completed and products generalized. This procedure is important when we want to update the ontology in the future.

7. Update and correct the ontology carefully according to maintaining guidelines and develop new enrichment and extension when a new domain is addressed.

In our research, we want to find the generalized association rules about the patterns in supermarket transactions, so we should build an ontology describing the supermar-
ket goods. As discussed above, the ontology is built by seven steps:

1. The domain and scope of our ontology is about the commercial wares in a supermarket. Since our experimental ontology is not very complex, we just make a brief plan about the procedures.

2. In step 2, after determining the potential end users of the ontology, we usually draw some tree-like graphs to give an outline. In our current research, the users are just the professors or students in our lab, so we do not care too much about the real meaning and just concentrate on the processes and algorithms. When applied to real projects, the end users are the managers or directors, so we must analyze the users’ requirements carefully and show some explicit drafts.

3. We have not found the existing ontologies, so we have to build a new one.

4. Based on the drafts in step 2, we define more details about the classes, relations and properties in the ontology. In our ontology, we just define is-a relationship, and a few properties, such as name, price, manufacturer, and so on. Due to the e-commerce web sites, we could easily collect enough terms and formalize them into the classes in the ontology. In fact, the items in these web sites are usually organized in a hierarchical structure, which could facilitate our work. However, we need modify it a lot to make it more suitable for the data mining.

5. We evaluate the ontology to see if it is reasonable, practical and useful. For more detailed evaluation, we should refer to some standards and test the ontology more completely.

6. Write some documents to record the important information about the ontology. Just like we write some documents for our computer programs, we should also write documents for ontology, which will be useful when we want to read and check it again. The documents include the drafts in Step 2, some general information in Step 4, and the evaluation results in Step 5. Of course, in our current research, the most important point is not the ontology building, but the ontology application and algorithm to be developed. So the documents in our research might not be so detailed.

7. Update it if necessary. Of course, we should log the changes into the documents.

3.4. Hierarchical Transaction

In our current research, the ontology is simplified to be a collection of classes and instances with is-a relationship, and serves as a conceptualized vocabulary to describe an application domain. In the ontology fragment in Fig. 2, each node denotes the abstract class or concrete instance, and each edge represents is-a relationship. The instance nodes will be used to replace the keyword attributes in traditional transaction databases and the hierarchy structure results in semantic relations to some extent. We still take the market-basket problem for example in this paper and build a shopping goods ontology by Protege, which is a free tool provided by Stanford University [8]. The items in the traditional transaction are replaced by instances in ontology, but the instances are not used for mining rules directly in this paper. The traditional transactions should be transferred into Hierarchical Transaction. In Hierarchical Transaction, the elements of each transaction consist of the classes in ontology instead of the instances. Let \( T = \{t_1, t_2, \ldots, t_i, \ldots, t_j\} \) be a set of transactions, and \( S = \{s_1, s_2, \ldots, s_j, \ldots, s_j\} \) be the instance set in the ontology, where \( I \) is the number of transactions and \( J \) is the number of instances, respectively. Each transaction, \( t_i = \{s_{i1}, s_{i2}, \ldots, s_{ik}, \ldots, s_{ik}\} \), is a subset of \( S \), where \( K_i \) is the number of goods in transaction \( t_i \). Let the set of classes in the ontology be denoted by \( C = \{c_1, c_2, \ldots, c_m, \ldots, c_M\} \), where \( M \) is the number of classes. There are is-a relations between instances and classes. That is, instance \( s_j \) belongs to a class and the super classes of this class, and we define these classes as Ancestor Classes of \( s_j \) denoted by \( \text{Ance}\{s_j\} \). Taking Fig. 2 for example, the Ancestor Classes of instance \( \text{Intelligent Running Shoes} \) are four classes: \( \text{Running Shoes}, \text{Men’s Shoes}, \text{Clothing & Shoes} \) and \( \text{Shopping Goods} \). The Hierarchical Transaction is denoted by \( T^\prime = \{t_{i1}, t_{i2}, \ldots, t_{i}, \ldots, t_{ij}\} \), and each element of Hierarchical Transaction is denoted as \( t_j = \{s_{j1}^1, s_{j2}^2, \ldots, s_{j2}^2, \ldots, s_{ij}^i\} \), whose element \( s_{jk}^i \) consists of the Ancestor Classes of the instance \( s_{jk} \), that is, \( s_{jk}^i = \text{Ance}\{s_{jk}\} \).

3.5. Quality Analysis of OGAR

After applying ontology to generalized association rules mining, we could get more new information from the data. In our current research, we only consider the classes, instances and relationships in an ontology, without considering the properties or facets. For example, after investigating the transactions data, we could find such association rules:

1. \( \text{Formal Dresses} \rightarrow \text{Dress Shoes} \)
   \( \text{(confidence: 0.73, support: 0.65)} \)

2. \( \text{Formal Dresses} \rightarrow \text{Women’s Shoes} \)
   \( \text{(confidence: 0.92, support: 0.96)} \)

3. \( \text{Formal Dresses} \rightarrow \text{Men’s Shoes} \)
   \( \text{(confidence: 0.51, support: 0.58)} \)

4. \( \text{Formal Dresses} \rightarrow \text{Men’s Running Shoes} \)
   \( \text{(confidence: 0.02, support: 0.07)} \)

From the first rule, we find that when a customer buys \( \text{Formal Dress} \), he or she may also buy \( \text{Dress Shoes} \) with high possibility. This rule suggests that the dress salesman should recommend some dress shoes to his customers.

From the second rule, we could see that the customer who has bought \( \text{Formal Dresses} \) will probably buy \( \text{Women’s Shoes} \) with very high possibility. This customer
may not buy *Dress Shoes*, but there is a very high probability that he or she will buy some *Women's Shoes*. As a result, the dress salesman should not only recommend the dress shoes but also some other women’s shoes to his dress buyers. If the salesman has already introduced some dress shoes to his dress buyer and the buyer shows little interest in them, the salesman should immediately introduce other kinds of women’s shoes (maybe with some discount).

From the third rule, it is known that the customer who has bought *formal dress* might also buy *men's shoes*. Usually the ladies wear formal dresses and attend the formal party together with gentlemen, so it is natural that the dress buyers may also buy some men’s clothes or shoes. We could conclude that the dress salesman should recommend some proper men’s shoes to his customers.

However, from the fourth rule, we find that the possibility is very low for a customer to buy *Formal Dresses* and *Men's Running Shoes* together. It means that if a customer has already bought the *Dress Shoes*, the salesman does not have to introduce *Men's Running Shoes* any more.

4. **Equalization of Classes from Different Layers**

If we mine rules among classes from all the layers, most of the items in rules may be from the upper layers, because the frequencies of the classes in the upper layers in the hierarchical transaction are much larger. We propose Dynamic Threshold Approach (DTA) for equalizing the classes from different layers in order to overcome the above problem. After the equalization, the items in association rules are picked up from different layers comparatively equally. First of all, some notations are defined to describe our method. In DTA, there is a different minimum support value in different layer $l$ defined by $SUP_{\text{min}}(l)$.

**Definition 1: $SUP_{\text{min}}(l)$**

$SUP_{\text{min}}(l)$ is the minimum support value in layer $l$. When we check a candidate rule $r$, e.g., $(C_1, C_2 \Rightarrow C_3, C_4)$, if all the classes in rule $r$ are from layer $l$, the support value of $r$, i.e., $support(r)$, should satisfy $Support(r) \geq SUP_{\text{min}}(l)$ if we want to store it into the rule pool.

If these classes in rule $r$ are from different layers, e.g., $l_1, l_2, l_3, l_4$, there are three methods to check the candidate rule $r$,

- maximum method:
  
  \[
  support(r) \geq \max\{SUP_{\text{min}}(l_1), SUP_{\text{min}}(l_2), SUP_{\text{min}}(l_3), SUP_{\text{min}}(l_4)\};
  \]

- minimum method:
  
  \[
  support(r) \geq \min\{SUP_{\text{min}}(l_1), SUP_{\text{min}}(l_2), SUP_{\text{min}}(l_3), SUP_{\text{min}}(l_4)\};
  \]

- average method:
  
  \[
  support(r) \geq \{SUP_{\text{min}}(l_1) + SUP_{\text{min}}(l_2) + SUP_{\text{min}}(l_3) + SUP_{\text{min}}(l_4)\}/4.
  \]

Of course, if there are more than 4 items in the candidate rules, we could easily extend the above methods.

In the following, $SUP_{\text{min}}$ represents the set of minimum support values in all the layers, that is,

\[
SUP_{\text{min}} = \{SUP_{\text{min}}(l_1), SUP_{\text{min}}(l_2), \ldots, SUP_{\text{min}}(l_L)\},
\]

where $L$ is the number or the set of suffixes of layers in the ontology. In order to calculate $SUP_{\text{min}}$, we define Candidate Class Rate ($CCR$) in Definition 2.

**Definition 2: CCR($l$)**

Given a certain layer $l$, $CCR(l)$ is defined as the ratio of the number of classes in layer $l$ whose support values are not less than $SUP_{\text{min}}(l)$. These classes could have the chance to be explored as the items in association rules while the other classes in this layer are deleted. For example, there are ten classes with support values in layer $l$ in Table 1, where the values in the left column are the original data without order and the values in the right column are sorted in descending order. If $CCR(l)$ is set at 0.5 ($l \in L$), $C_4, C_{10}, C_9, C_8, C_2$ would have the chance to appear in the association rules and the other five classes are excluded. $SUP_{\text{min}}$ should be 0.57 in order to select $C_4, C_{10}, C_9, C_8, C_2$. In the following of this paper, $CCR$ represents the set of ratio values in all the layers, i.e., $CCR = \{CCR(l_1), CCR(l_2), \ldots, CCR(l_L)\}$. The main ideas of DTA are explained by Algorithm 3.1.

The notations in the algorithm are as follows:

- $F_l$: the set of support values of all classes in layer $l$ with order, $F_l = \{f_{l1}, f_{l2}, \ldots, f_{lM_l}\}$;
- $M_l$: the number of classes in layer $l$;
- $\text{integer}(x)$: get the integer part of the numerical value $x$;
- $support(r)$: the support value of rule $r$;
- $\text{confidence}(r)$: the confidence value of rule $r$;
- $CONF_{\text{min}}$: the minimum confidence value;
- $\text{layer}(c)$: get the layer of class $c$ in ontology;
- $\text{store}(r)$: store rule $r$ into the rule pool.

**Algorithm 4.1: DTA($r$)**

\[
\text{for each } l \in L \\
\text{do } \{ \quad \text{s } \leftarrow \text{integer}(M_l \times CCR(l)) \\
\text{ } \{ \quad \text{SUP}_{\text{min}}(l) = f_{ls} \\
\text{ } \} s \leftarrow 1 \\
\text{for each class } c \text{ in rule } r \\
\text{do } \{ \quad l \leftarrow \text{layer}(c) \\
\text{ } \{ \quad \text{if } SUP_{\text{min}}(l) < s \\
\text{ } \} \text{ then } s \leftarrow SUP_{\text{min}}(l) \\
\text{if support($r$)} \geq s \text{ and confidence($r$)} \geq CONF_{\text{min}} \text{ then } \text{store($r$)}
\}
\]

In Algorithm 3.1, we first calculate $SUP_{\text{min}}(l)$ by $CCR(l)$. Then, we compare the $SUP_{\text{min}}(l)$s, where layer $l$ includes the classes of candidate rule $r$ and select the minimum one as the minimum support to check the rule $r$. In this algorithm we apply the minimum method. If we employ the average method, we would get fewer association rules, and if we make use of the maximum method, we would find the fewest rules.
Table 1. Support values of classes in layer l.

<table>
<thead>
<tr>
<th>Class</th>
<th>Support value</th>
<th>Class</th>
<th>Support value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C_1</td>
<td>0.30</td>
<td>C_4</td>
<td>0.88</td>
</tr>
<tr>
<td>C_2</td>
<td>0.57</td>
<td>C_{10}</td>
<td>0.85</td>
</tr>
<tr>
<td>C_3</td>
<td>0.12</td>
<td>C_9</td>
<td>0.72</td>
</tr>
<tr>
<td>C_4</td>
<td>0.88</td>
<td>C_8</td>
<td>0.59</td>
</tr>
<tr>
<td>C_5</td>
<td>0.53</td>
<td>C_2</td>
<td>0.57</td>
</tr>
<tr>
<td>C_6</td>
<td>0.40</td>
<td>C_5</td>
<td>0.53</td>
</tr>
<tr>
<td>C_7</td>
<td>0.37</td>
<td>C_6</td>
<td>0.40</td>
</tr>
<tr>
<td>C_8</td>
<td>0.59</td>
<td>C_7</td>
<td>0.37</td>
</tr>
<tr>
<td>C_9</td>
<td>0.72</td>
<td>C_1</td>
<td>0.30</td>
</tr>
<tr>
<td>C_{10}</td>
<td>0.85</td>
<td>C_3</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Fig. 4. Directed Graph Structure

5. Ontology Based Generalized Association Rules Mining

5.1. Genetic Network Programming

Genetic Network Programming (GNP) is an extension of GP, which uses directed graphs as genes for evolutionary computation [10, 11]. As Fig. 4 shows, the basic structure of GNP has three kinds of nodes: a Start Node, some Judgement Nodes and some Processing Nodes. The start node is a special node, indicating the first position to be executed. Judgement nodes receive the information from the environment and determine the next node to be executed. Processing nodes represent some functional actions that could be taken. GNP evolves the graph structure with the predetermined number of nodes, and reuses these judgement nodes and processing nodes during the execution. As a result, it could be quite compact and efficient and never cause the bloat.

5.2. Mining OGAR with GNP

When applying GNP to data mining, it is predigested into a kind of chain structure. We first present the most basic structure, i.e., Sequentially Connected Chain (SCC) in Fig. 5(a) to show the principle of our method.

SCC is a chain of judgement nodes, e.g. A=1, B=1, C=1, etc, which consist of the classes in ontology where each judgement node has only one previous judgement node and one next judgement node. SP is the start node and EP is the end node representing the end of execution. Each judgement node has a corresponding processing node, for example, P_1 for A=1 and P_2 for B=1. Judgement nodes, like A=1, B=1, C=1, etc, decide whether the class A, B or C occurs in the Hierarchical Transaction when scanning the transactions. If the judgement result is negative, the corresponding processing nodes will be activated, otherwise the program will go on to the next judgement node. For example, if C is not in transaction t_i, P_3 will be activated to stop the scan and check the rule, and if C occurs, the program will continue to check D without dealing with P_3. As soon as one processing node is activated, it will check the candidate association rules just scanned.

Here is a simple example in Fig. 5(a) and Table 2 to show the process of using SCC to find association rules. Without loss of generality, we assume that there are 10 records of transactions in the database. Class A occurs 8 times in the records, class B occurs 5 times in the records with the occurrence of A, and class C occurs 3 times in the records with occurrence of A and B. If class D does not occur in the records with A, B and C, which means the judgment result of D=1 is negative, the processing node P_4 is activated to end the scanning and the support and confidence value of the candidate rules are calculated. If we set 40% for the minimal support value and 60% for the minimal confidence value, we could get one association rule (A \Rightarrow B) from the candidate rules in Table 2.

We specify a parameter, Maximum Rule Length (MRL), to restrict the rule length, because sometimes even the short length rules contain enough information for us and some long length rules may be redundant. Of course, when MRL is the length of SCC, that is, MRL equals to the number of judgement nodes in SCC, there is no restriction for rule length.

Table 2. An example of candidate rules.

<table>
<thead>
<tr>
<th>Candidate rules</th>
<th>Support</th>
<th>Confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>A,B \Rightarrow C</td>
<td>30%</td>
<td>60%</td>
</tr>
<tr>
<td>A \Rightarrow B,C</td>
<td>30%</td>
<td>38%</td>
</tr>
<tr>
<td>A \Rightarrow B</td>
<td>50%</td>
<td>63%</td>
</tr>
</tbody>
</table>
5.3. Improved Item Chain

In the above sections, SCC is introduced to show the basic principles of our method. However, SCC is not powerful enough to exploit the potential search ability of GNP because of its simple structure. So we design an improved structures: Randomly Connected Chain (RCC).

The items in RCC are not only selected randomly, but also connected randomly, which could have one next item and multiple previous items. Although there are some possible loops in RCC, e.g., the loop between A and D in Fig. 5(b), its random connections may be effective to find more rules than SCC. Maximum Rule Length (MRL) could be used to end this loop.

Another potential improved approach is to use multiple Start Nodes for SCC or RCC.

We have compared the performances of different item chains in the previous research [12] and it has been proved that RCC could have better performance than SCC. We employ RCC, which could also represent more features of GNP, to evaluate the methods proposed in this paper.

5.4. Fitness Function

There are two kinds of fitness functions defined in our method: Fitness of Individual ($F_i$) and Fitness of Class ($F_c$).

We first define One Step Reward (OSR) to make use of the reward difference between the current generation and the previous generation.

**Definition 3: OSR**

After executing the $(g - 1)$th generation, suppose $M^c(g - 1)$ rules containing class $c$ are produced. Then, after the $g$th generation, suppose $M^c(g)$ rules containing class $c$ are produced. We call $\Delta M^c(g) = M^c(g) - M^c(g - 1)$ as One Step Reward (OSR) of class $c$.

**Definition 4: Fitness of individual denoted by $F_i$**

$F_i$ is the fitness of individuals used for selection.

We introduce four methods to select individuals by calculating the fitness value of individual $i$ in the $g$th generation $F_i(g)$: Random Policy Method, Number Policy Method, OSR Policy Method and Hybrid Policy Method.

1. **Random Policy Method**

   We select individuals randomly to generate the next generation.

2. **Number Policy Method**

   When applying Number Policy Method, the fitness value of individual $i$ in the $g$th generation is described by Eq. (1):

   $$F_i(g) = N_i(g) \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots (1)$$

   where $N_i(g)$ is the number of rules individual $i$ produces in the $g$th generation.

3. **OSR Policy Method**

   When applying OSR Policy Method, $F_i$ of individual $i$ is defined as:

   $$F_i(g) = \sum_{c \in C_i} \Delta M^c_i(g) \quad \ldots \ldots \ldots \ldots \ldots \ldots (2)$$

   $$\Delta M^c_i(g) = \begin{cases} M^c_i(g), & g = 1; \\ M^c_i(g) - M^c_i(g - 1), & g > 1. \end{cases} \quad (3)$$

   where $M^c_i(g)$ is the number of rules containing class $c$ that individual $i$ produces in the $g$th generation, and $C_i$ is the set of classes in the rules that individual $i$ produces. $\Delta M^c_i(g)$ is OSR value.

4. **Hybrid Policy Method**

   If $N_{max}(g) \neq N_{min}(g)$,

   $$F_i(g) = \frac{N_i(g) - N_{min}(g)}{N_{max}(g) - N_{min}(g)} \times \left( \frac{\arctan{\sum_{c \in C_i} \Delta M^c_i(g)}}{\pi} + \frac{1}{2} \right) \quad \ldots \ldots \ldots \ldots \ldots \ldots (4)$$

   where $N_{min}(g) = \min_{i \in I} N_i(g), N_{max}(g) = \max_{i \in I} N_i(g), I$ is the set of suffixes of individuals.

   It could be easily inferred that $0 < F_i(g) < 1$.

   If $N_{max}(g) = N_{min}(g) = 0, F_i(g) = 0$.

   If $N_{max}(g) = N_{min}(g) \neq 0, F_i(g) = 1$.

**Definition 5: Fitness of class denoted by $F_c$**

$F_c$ is the fitness of classes used for the selection of classes.

There are also four methods to select the classes by calculating the fitness value $F_c$ of class $c$: Random Policy Method, Number Policy Method, OSR Policy Method and Hybrid Policy Method.

1. **Random Policy Method**

   We select classes randomly.

2. **Number Policy Method**

   When applying Number Policy Method, $F_c$ of class $c$ in the $g$th generation is

   $$F_c(g) = M^c(g) \quad \ldots \ldots \ldots \ldots \ldots \ldots (5)$$

   where $M^c(g)$ is the number of rules containing class $c$ produced in the $g$th generation.

3. **OSR Policy Method**

   When applying OSR Policy Method, the $F_c$ of class $c$ in the $g$th generation is defined as:

   $$F_c(g) = \Delta M^c(g) \quad \ldots \ldots \ldots \ldots \ldots \ldots (6)$$

   $$\Delta M^c(g) = \begin{cases} M^c(g), & g = 1; \\ M^c(g) - M^c(g - 1), & g > 1. \end{cases} \quad (7)$$

   where $\Delta M^c(g)$ is called OSR value.
dividuals. If the classes are chosen by renewal operator 
say that they are brand new individuals to a certain extent. 
contains much fresher classes and connections, so we could 
the new GNP individuals formed by renewal operator con-
dividuals generated by crossover and mutation operators, 
form the new individuals. Compared with the new in-
5.5. Genetic Operators

We develop a new genetic operator named renewal op-
erator in this paper. Renewal operator selects classes from 
the ontology class library according to their $F_c$ values to 
form the new individuals. Compared with the new in-
dividuals generated by crossover and mutation operators, 
the new GNP individuals formed by renewal operator con-
tains much fresher classes and connections, so we could 
say that they are brand new individuals to a certain extent. 
This process is similar to the creation of the initial in-
dividuals. If the classes are chosen by renewal operator 

(4) Hybrid Policy Method
If $M_{\text{max}}(g) \neq M_{\text{min}}(g)$,
$$F_c(g) = \frac{M^c(g) - M_{\text{min}}(g)}{M_{\text{max}}(g) - M_{\text{min}}(g)} \times \left( \frac{\arctan M^c(g)}{\pi} + \frac{1}{2} \right)$$

where $M_{\text{min}} = \min_{c \in C} M^c(g)$, $M_{\text{max}} = \max_{c \in C} M^c(g)$, and $C$ is the 
set of classes in the rules produced by all the individuals 
in the $g$th generation.

We could infer that $0 \leq F_c(g) \leq 1$.
If $M_{\text{max}} = M_{\text{min}} = 0$, $F_c(g) = 0$.
If $M_{\text{max}} = M_{\text{min}}$ $\neq 0$, $F_c(g) = 1$.

Although our method does not guarantee to find all the 
rules, we still want to mine rules as many as possible.
OSR Policy, Number Policy, and Hybrid Policy are pro-
ted to give the priority on the important individuals and 
classes, so that these individuals and classes may give us 
more rules in the later generations.

However, different policies have different criterions to 
judge whether an individual or a class is important or not.
- When Number Policy is used, the selected important 
individual or class is the one which produces more 
rules in $n$th generation.
- When OSR Policy is used, the selected important in-
dividual or class is the one which has the higher pro-
ducing speed of the rules.
- When Hybrid Policy is applied, we combine the 
above two policies.

If an individual is more important, it will have more 
chance to be selected by crossover or mutation operators. 
If a class is more important, it will have more possibility 
to be selected by renewal operator.

We will compare the effectiveness of these policies in 
Section 6.1 and Section 6.2.

5.6. Meaningfulness Assurance

When we say that a rule is meaningful, it means that 
this rule cannot be deduced directly by other rules. For 
example, if rule $(A \Rightarrow B)$ exists and $B$ belongs to $\text{Ance}(A)$, 
this rule is not meaningful, because it can be simply de-
duced from the fact that $B$ belongs to $\text{Ance}(A)$. We could 
ensure the meaningfulness by two kinds of methods: pre-
check and post-check.

Pre-check is to make sure that all rules are meaning-
ful when generating the individuals. When generating the 
individuals, if we select a judgement node from layer $l$, 
the next judgement node will not be from the layer higher 
than $l$. For example, in Fig. 5(a), if node A of an in-
dividual is in the layer 3 of ontology, the next node B 
should be selected from layer 3, 4, 5 or 6. However, pre-
check may probably miss some rules. For example, in 
Fig. 2, we will never find rules like 52% customers who 
buy women’s shirts may also purchase pets. Genetic op-
erators may also destroy such assurance unless we give 
some additional limitations when performing genetic op-
erations.

Post-check is to ensure the meaningfulness by deleting 
the meaningless part of the rules when checking the can-
didate rules. For example, in Fig. 5(a), when checking 

randomly, this process is almost the same as the initial 
process. However, the renewal has considered the $F_c$ cal-
culated by Eqs. (5), (6) and (8) which could not be con-
sidered at the initial stage.

The reason for introducing renewal operator is that 
we want to find useful individuals as many as possible.
Crossover and mutation could find many rules in a local 
search space, but they are not efficient enough to explore 
global search space. We introduce renewal operator to let 
some individuals jump to the brand new search space im-
mediately. Compared with other methods, Random Pol-
icy Method produces the newer search spaces, because it 
chooses classes completely randomly, and does not con-
sider the search space which has already been searched.
However, it is not true that the newer search space has the 
better performance. We prove in Section 6.1 that using the 
other methods with feedback, such as OSR Policy Method 
and Number Policy Method, produces better results.
the candidate rule \((A, B \Rightarrow C)\), if \(B\) is an ancestor class of \(A\), we can delete \(B\) from the candidate rule and check a new candidate rule \((A \Rightarrow C)\). Of course, the frequency of the classes in this new candidate rule should be counted again. There are three procedures in post-check:

First, we check the antecedent part of the candidate rule to ensure that no item is the Ancestor Class of the other items in this part.

Second, we check the consequent part to ensure that no item is the Ancestor Class of the other items in this part.

Third, we make sure that no item in the consequent part is the Ancestor Class of the items in the antecedent part.

5.7. Hash Function for Avoiding Redundance

As soon as we find some rules, the newly found rules should be compared with the existing rules in order to avoid redundancy. Suppose there are \(n\) rules already in the pool with an average number of \(m\) classes. The simplest method is that we compare every class of the newly mined rule with every class of all the existing rules, and then the comparison time would be about \(n \times m\). We will discuss another two hash function methods for much faster comparison.

(1) Hash Method I

In this method, each rule has a corresponding hash value by hash function \(h(r)\). The hash value of rule \(r\) is calculated by:

\[
h(r) = \sum_{c \in C} r_c \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (9)
\]

where \(r_c\) is the unique ID of class \(c\) in the rule \(r\) and \(C\) is the set of classes in the rule \(r\). When a new candidate rule \(r\) is found, we calculate the hash value \(h(r)\) of this rule and compare \(h(r)\) with all the hash values stored in the hash table one by one. There is a two-dimensional array, where the elements in the first dimension are the hash values, and the elements in the second dimension are the rules with the same hash values. If we cannot find any existing hash value equal to \(h(r)\), \(r\) is a new rule. Otherwise, we compare the candidate rule \(r\) with all the rules having the same hash value \(h(r)\) and \(r\) is a new rule only if \(r\) is different from all these rules. The time for comparison is about \(n + am\), where \(a\) is a parameter describing the number of collisions with the same hash value.

(2) Hash Method II

This method makes use of an improved hash function \(H(r)\) defined as:

\[
H(r) = \sum_{c \in C} r_c \mod 1000 \quad \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (10)
\]

where \(r_c\) is the unique ID of class \(c\) in rule \(r\). There is also a two-dimensional hash array storing hash values and collisions. The size of the first dimension is 1000. The hash values are the index of the first dimension. The elements in the first dimension are the number of collisions and the elements in the second dimension are the rule strings with the same hash value. The rule string of \(r\), \(str(r)\), is represented in the following form:

\[
str(r) = "sp + \alpha_1 + \cdots + \alpha_p, sq + \beta_1 + \cdots + \beta_q" \quad (11)
\]

where \(p\) is the number of antecedent classes, \(q\) is the number of consequent classes, \(\alpha\) is the ID of antecedent classes, and \(\beta\) is the ID of consequent classes. When a new rule \(r\) is mined, if the \(H(r)\)th element of the first dimension is 0, this means that \(r\) is a new rule. Otherwise, we should compare \(str(r)\) with all the collision rule strings. The comparison time of this hash method is about \(1 + b\), where \(b\) is a parameter describing the number of collisions. We give an example to show the comparison process by Hash Method II in Fig. 7. If the hash value of rule \(r\) is 572, the corresponding element of the first dimension is 3, which means that there are 3 collisions with the same hash value 572. Then we compare \(str(r)\) with all the collision rule strings. If the hash value of rule \(r\) is 574, the 574th element of the first dimension is 0 and this indicates that \(r\) is a new rule. Then, we add rule \(r\) into the rule pool, increase the 574th first dimension value by 1 and store \(str(r)\) into the second dimension. Hash Method II could exceed the two other methods greatly when \(n\) becomes very large, although this advantage may not be obvious when \(n\) is small. For example, if \(n=5, m=2, a=2\) and \(b=2\), the comparison time for the method without hash, Hash Method I and Hash Method II are 10, 9, 3 respectively. But when \(n\) becomes 100, the time are 200, 104 and 3 respectively. We apply Hash Method II in our evaluations.

6. Experiments and Results

The testing datasets for evaluation are synthetic. If we select classes randomly from the ontology to generate the transactions, we could mine merely few rules and these transactions cannot mimic the real-world situations. In real situations, some items occur frequently in the transactions while the other items seldom occur. Take supermarket for example, people buy foods and drinks much more frequently than televisions and computers. So we make a small part of the classes in ontology to generate most of the classes in the transactions. For example, if
Table 3. Some parameters for the following experiments.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT</td>
<td>Number of transactions</td>
<td>1000</td>
</tr>
<tr>
<td>NC</td>
<td>Number of classes</td>
<td>364</td>
</tr>
<tr>
<td>NL</td>
<td>Number of layers in ontology</td>
<td>5</td>
</tr>
<tr>
<td>NInd</td>
<td>Number of individuals</td>
<td>10</td>
</tr>
<tr>
<td>JInd</td>
<td>Number of judgment nodes in each individual</td>
<td>10</td>
</tr>
<tr>
<td>NS</td>
<td>Number of start nodes</td>
<td>5</td>
</tr>
<tr>
<td>NG</td>
<td>Number of generations</td>
<td>10000</td>
</tr>
<tr>
<td>Pr</td>
<td>Crossover rate</td>
<td></td>
</tr>
<tr>
<td>Ps</td>
<td>Renewal rate</td>
<td></td>
</tr>
<tr>
<td>Pcm</td>
<td>Mutation rate for content</td>
<td>0.1</td>
</tr>
<tr>
<td>Pcn</td>
<td>Mutation rate for connection</td>
<td>0.1</td>
</tr>
<tr>
<td>CONFmin</td>
<td>Minimum confidence value</td>
<td>0.5</td>
</tr>
<tr>
<td>Smin</td>
<td>Minimum support value (without equalization)</td>
<td>0.5</td>
</tr>
<tr>
<td>MRL</td>
<td>Max rule length</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4. The results of Experiment 1 where each result is obtained using different Fi and Fc policy combination. We also calculate the average values in each column and each row. From the average column, we could compare the overall performance of different policies for Fi, while from the average row we could compare the overall performance of different policies for Fc.

<table>
<thead>
<tr>
<th>Fi</th>
<th>Fc</th>
<th>R</th>
<th>N</th>
<th>O</th>
<th>H</th>
<th>avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R+R</td>
<td></td>
<td>279.1</td>
<td>280.7</td>
<td>309.5</td>
<td>275.3</td>
<td>286.2</td>
</tr>
<tr>
<td>N+R</td>
<td></td>
<td>346.3</td>
<td>332.2</td>
<td>388.9</td>
<td>353.8</td>
<td>355.3</td>
</tr>
<tr>
<td>O+R</td>
<td></td>
<td>299.8</td>
<td>309.5</td>
<td>340.4</td>
<td>304.7</td>
<td>313.6</td>
</tr>
<tr>
<td>H+R</td>
<td></td>
<td>329.2</td>
<td>347.8</td>
<td>358.4</td>
<td>333.2</td>
<td>342.2</td>
</tr>
<tr>
<td>avg.</td>
<td></td>
<td>316.6</td>
<td>317.6</td>
<td>349.3</td>
<td>316.8</td>
<td></td>
</tr>
</tbody>
</table>

6.1. Experiment 1

The first experiment is designed to test the effectiveness of different policies of Fi and Fc. We test the performances of the combinations of different Fi and Fc policies. Since there are 4 kinds of Fi policies and 4 kinds of Fc policies, so we have totally 16 kinds of testing combinations, i.e., R+R, R+N, R+O, R+H, N+R, N+N, N+O, N+H, O+R, O+N, O+O, O+H, H+R, H+N, H+O, H+H (R: Random Policy, N: Number Policy, O: OSR Policy, H: Hybrid Policy), where R+R means Fi employs Random Policy and Fc uses Random Policy, R+N means Fi employs Random Policy and Fc uses Number Policy, and so on. Each combination is tested for 10 times and we calculate the average results. We apply DTA with what is the best ratio of crossover and renewal.

Considering the results of Experiment 1, Number Policy is chosen for Fi and OSR Policy is chosen for Fc to test the experiment. We design 5 kinds of tests with and without equalization using the parameters set in Table 3.

Table 5. Rates of crossover and renewal used in Experiment 2.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Pcm</th>
<th>Pcn</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0R8</td>
<td>0.0</td>
<td>0.8</td>
</tr>
<tr>
<td>C2R6</td>
<td>0.2</td>
<td>0.6</td>
</tr>
<tr>
<td>C4R4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>C0R2</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>C8R0</td>
<td>0.8</td>
<td>0.0</td>
</tr>
</tbody>
</table>

6.2. Experiment 2

In this experiment, we test whether the renewal operator could really speed up the rule searching process and what is the best ratio of crossover and renewal.

From both Figs. 8(a) and (b), we can see that renewal could improve the effectiveness and efficiency remarkably. However, it is not true that the more renewal used, the better performance could be gotten. The tradeoff point occurs near C2R6, although the performance of C4R4 seems to be very close to that of C2R6.

Although the different policies for Fi and Fc in Experiment 1 and the renewal operator in Experiment 2 may reflect the eventual number of rules, the computational costs for these experiments are almost the same. Almost all the experiments need about 180 seconds to finish 10000 generations.

6.3. Experiment 3

We design this experiment to study the effect of equalization. Number Policy for Fi and OSR Policy for Fc are also chosen in the experiment, and the best result C2R6 in Experiment 2 is adopted.

Figure 9 shows the layer distribution of the classes in all the rules in the 10000th generation, where Fig. 9(a) shows the results when applying DTA and Fig. 9(b) shows the results when not applying DTA. The layer distribution is the statistic of how many times the class in the rules
A Genetic Network Programming Based Method to Mine Generalized Association Rules with Ontology

belongs to each layer. For example, if there are three rules, (A, B ⇒ C), (C, E ⇒ H) and (J ⇒ C), where class A, B and C are in layer1, class E and H are in layer2, and class J is in layer3. Then the layer distribution result is like this, layer1 is assigned with a value 5, layer2 with 2, and layer 3 with 1. The experiment was carried out 10 times.

In Fig. 9(a), classes in the rules are from each layer fairly in balance except layer 1. However, from Fig. 9(b) we can see that all the classes in the rules are from layer1 and layer2, without any class from the other layers, when not equalizing the layers. Because when using DTA each element of $CCR$ is set at 0.3 and the number of classes in the layer1 is 3, as a result, $3 \times 0.3 = 0.9 < 1$, therefore, none of the classes from layer1 is selected to form the rules. Of course, if we change the each element of $CCR$, e.g., 0.4, since $3 \times 0.4 = 1.2 > 1$, then one class from the layer1 will occur in the rules.

However, we set a single value 0.3 to all the $CCR(l)$, then it is shown in Fig. 9(a) that the concepts in the second and fifth layers occur more frequently than the other layers. In fact, we could also set different $CCR$ value to different layers to control the balance of the appearance of the concepts from different layers. For example, if we set $CCR(l)$=[0.5, 0.2, 0.5, 0.5, 0.1], the results of the experiment become like Fig. 10, and we could see that the layers could reach a certain dynamic balance by adjusting the $CCR$ values. In fact, the absolute balance, i.e., resulting in almost the same number of rules in all the layers, could not always be the best choice, because the user may want to mine more rules in some of the layers. As a result, our method with dynamic balance would be a suitable way of obtaining rules on ontology.

6.4. Experiment 4

This experiment studies the number of rules when using different $CCR$ value which is unique for all the layers. Fig. 11 shows the result when changing $CCR$ from 0.2 to 0.6. From Fig. 11, it is obvious that when $CCR$ becomes
larger, much more rules are produced. It is also shown that if $CCR$ increases by 0.1, the number of rules in the last generation increases by 1.5-3 times. If $CCR$ is set at more than 0.6, i.e., 0.7-0.9, more than 10000 rules are obtained.

6.5. Experiment 5

We evaluate and compare the different redundancy checking methods, i.e., No Hash Method, Hash Method I, Hash Method II, in this experiment. Fig. 12 shows the experimental results.

From Fig. 12, we could see that Hash Method II is the fastest one, while No Hash Method is the slowest one, which also proves the analysis in Section 5.7.

7. Conclusion

In this paper, we make use of ontology to extend the association rules mining in the multi concept layers and introduce Dynamic Threshold Approach to equalize the different layers of ontology. We also apply Genetic Network Programming to mine rules generation by generation. The experiments show that DTA could efficiently equalize the classes from different layers and the number of rules increases by 1.5-3 times when $CCR$ increases by 0.1. The experimental results also show that the new renewal operator could improve the performance remarkably. In addition, it is shown that Number Policy is the best policy for $F_i$ and OSR Policy is the best one for $F_c$ among the four policies we proposed.

8. Future Work

In this paper, we only consider a part of the ontology for data mining and properties and facets of ontology are almost ignored. In fact, there are much more knowledge contained in the properties and facets in the ontology. In our future work, we will continue the following research:

1. Consider the properties and facets of the ontology. For example, if we consider the trademark property and color property, we may find such kind of rules:

   - IBM Computer $\rightarrow$ White SONY Earphone (confidence:0.23, support:0.20)
   - IBM Computer $\rightarrow$ Black SONY Earphone (confidence:0.35, support:0.31)
   - IBM Computer $\rightarrow$ Red SONY Earphone (confidence:0.02, support:0.07)

   From the above rules, we could find that Black SONY Earphone is the best tie-in for IBM Computer.

2. Consider the numeric values of items. For example, we could add purchase price properties, then we may find the following rules:

   - Formal Dress priced at more than $1200$ $\rightarrow$ Dress Shoes priced at more than $1850$ (confidence:0.47, support:0.36)
   - Formal Dress priced at more than $1200$ $\rightarrow$ Dress Shoes priced at less than $50$ (confidence:0.05, support:0.04)
   - Formal Dress priced at less than $100$ $\rightarrow$ Dress Shoes priced at less than $50$ (confidence:0.60, support:0.35)

   After analyzing the above rules, we could recommend the goods to the customers more properly.

As the future work, we will also apply our method to some real data in a specific domain.

References:


A Genetic Network Programming Based Method to Mine Generalized Association Rules with Ontology


Name: Guangfei Yang
Affiliation: Ph.D. Student, Graduate School of Information, Production and Systems, Waseda University
Address: 2-7 Hibikino, Wakamatsu-ku, Kitayushu, Fukuoka 808-0135, Japan
Brief Biographical History: 2005 Received the Master degree from Dalian University of Technology, China 2006- Ph.D. student at Graduate School of Information, Production and Systems, Waseda University
Main Works:

Name: Kaoru Shimada
Affiliation: Dr., Visiting Lecturer, Information, Production and Systems Research Center, Waseda University
Address: 2-7 Hibikino, Wakamatsu-ku, Kitayushu, Fukuoka 808-0135, Japan
Brief Biographical History: 1999 Received M. Med. Sci. degree from University of Tsukuba 2007 Received Ph.D. degree from Graduate School of Information, Production and Systems, Waseda University 2007- Visiting Lecturer at Waseda University
Main Works:

Membership in Academic Societies:
- The Society of Instrument and Control Engineers (SICE)
- The Information Processing Society of Japan (IPSJ)
- The Institute of Electrical Engineers of Japan (IEEJ)

Name: Shingo Mabu
Affiliation: Dr., Assistant professor, Graduate School of Information, Production and Systems, Waseda University
Address: 2-7 Hibikino, Wakamatsu-ku, Kitayushu, Fukuoka 808-0135, Japan
Brief Biographical History: 2003 Received the M.E. degree from Kyushu University 2006 Received Ph.D. degree from Graduate School of Information, Production and Systems, Waseda University 2006- Visiting Lecturer at Waseda University 2007- Assistant Professor at Waseda University
Main Works:

Membership in Academic Societies:
- Institute of Electrical and Electronics Engineers (IEEE)
- The Society of Instrument and Control Engineers (SICE)
- The Institute of Electrical Engineers of Japan (IEEJ)
Name: Kotaro Hirasawa
Affiliation: Dr., Professor, Graduate School of Information, Production and Systems, Waseda University

Address: 2-7 Hibikino, Wakamatsu-ku, Kitayushu, Fukuoka 808-0135, Japan

Brief Biographical History:
1966 Joined Hitachi Corp.
1992- Professor at Kyushu University
2002- Professor at Waseda University

Main Works:

Membership in Academic Societies:
- Institute of Electrical and Electronics Engineers (IEEE)
- Association for Computing Machinery (ACM)
- The Society of Instrument and Control Engineers (SICE)
- The Institute of Electrical Engineers of Japan (IEEJ)
- The Information Processing Society of Japan (IPSJ)

Name: Jinglu Hu
Affiliation: Dr., Associate Professor, Graduate School of Information, Production and Systems, Waseda University

Address: 2-7 Hibikino, Wakamatsu-ku, Kitayushu, Fukuoka 808-0135, Japan

Brief Biographical History:
1997 Received Ph.D. degree from Kyushu Institute of Technology
1997- Research Associate at Kyushu University
2003- Associate Professor at Waseda University

Main Works:

Membership in Academic Societies:
- The Society of Instrument and Control Engineers (SICE)
- The Institute of Electrical Engineers of Japan (IEEJ)